Automorphisms of transition graphs for a linear cellular automaton

Edward J. Powley and Susan Stepney

Department of Computer Science, University of York, UK

Abstract. A *cellular automaton (CA)* is a discrete dynamical system, and the *transition graph* is a representation of the CA's phase space. *Automorphisms* of the transition graph correspond to symmetries of the phase space; studying how the total number of automorphisms varies with the number of cells on which the CA operates yields a (partial) classification of the space of CA rules according to their dynamical behaviour.

In the general case, to find the number of automorphisms we must iterate over the entire transition graph; thus the time complexity is exponential with respect to the number of cells. However, if the CA is *linear*, the transition graph has properties which allow the number of automorphisms to be computed much more efficiently. In this paper, we investigate the numbers of automorphisms for a particular linear CA, *elementary rule 90*. We observe a relationship between the number of automorphisms and a number theoretical function, the *suborder function*.

1 Introduction

A *cellular automaton (CA)* consists of a finite nonempty set of *states*, a discrete lattice of *cells*, and a *local update rule* which maps deterministically the state of a cell and its neighbours at time t to the state of that cell at time $t + 1$. A *configuration* of a CA is an assignment of a state to each cell. The local update rule extends to a *global map*, a function from configurations to configurations, in the natural way.

The *transition graph* of a CA is a directed graph whose vertices are the configurations of the CA, and whose edges are determined by the global map. There is an edge from vertex r to vertex s if and only if the global map sends configuration r to configuration s. The transition graph is a representation of the overall structure of the phase space of the CA: in particular, the *automorphisms* (self-isomorphisms or "symmetries") of the transition graph are, in a sense, the symmetries of the CA's dynamics [1]. Examples of transition graphs are shown in Figs. 1 and 2.

In [1], we investigate how numbers of automorphisms vary with the number N of cells on which the CA operates (Fig. 3). For the majority of CA rules, there seems to be a linear relationship between the number of automorphisms and e^{e^N}.

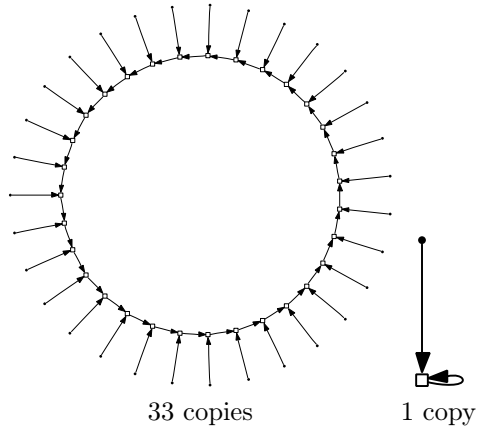

Fig. 1. Transition graph for rule 90 on 11 cells.

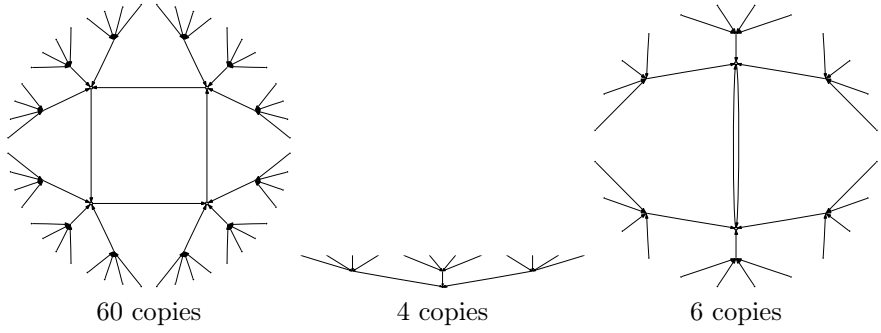

Fig. 2. Transition graph for rule 90 on 12 cells.

However, we identify two classes of CAs for which this linear correspondence does not seem to hold. One of these classes consists almost entirely of *linear CAs* (CAs whose local update rule is a linear function), and is characterised by the "zig-zag" pattern depicted in Fig. 4 (a). In this paper, we investigate this pattern more closely.

As is the case with many other classes of system, linear CAs submit much more readily to analysis than their non-linear counterparts. Indeed, the operation of a linear CA is simply repeated convolution of a configuration with a fixed sequence corresponding to the rule, which in turn is equivalent to repeated multiplication in a finite ring of polynomials. Martin et al [2] use this fact to study linear CAs, and succeed in proving several results about one linear CA in particular (*elementary rule 90*, in Wolfram's terminology [3]). We use these results to derive an algorithm, dramatically more efficient than the general al-

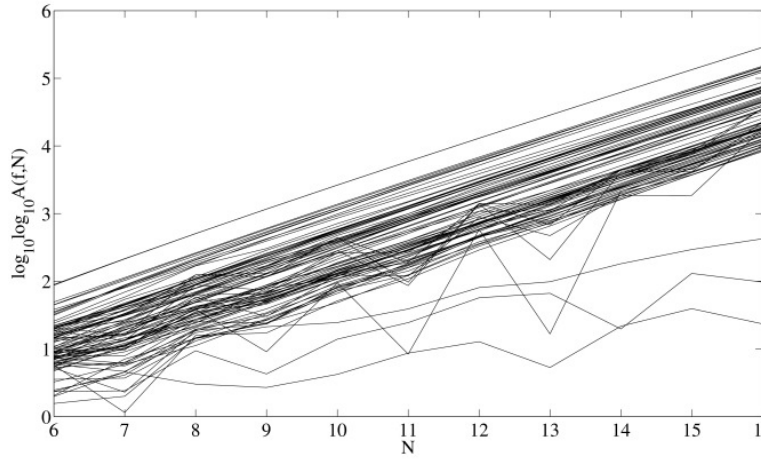

Fig. 3. Plot of $\log_{10} \log_{10} A(f, N)$ (where $A(f, N)$ is the number of automorphisms) against N, for $6 \leq N \leq 16$ and for all 88 essentially different ECA rules. From [1].

gorithm described in [1], for computing the number of automorphisms for the transition graphs of rule 90.

We argue, but do not prove, that the "zig-zag" oscillations in the number of automorphisms for rule 90 on N cells correspond to the oscillations of a number theoretical function known as the *suborder function*.

2 Linear CAs and polynomials

We restrict our attention to finite one-dimensional CAs, i.e. we take the lattice to be $\mathbb{Z}_N$ (the cyclic group of integers modulo N). This lattice has *periodic boundary condition*, in that we consider cell $N-1$ to be adjacent to cell 0. The neighbourhood is specified in terms of its *radius* r, so that the neighbours of cell i are cells $i-r, \ldots, i+r$. We further restrict our attention to CAs whose state set is also a cyclic group, say $\mathbb{Z}_k$. Thus the local update rule is a function $f : \mathbb{Z}_k^{2r+1} \to \mathbb{Z}_k$, which extends to a global map $F : \mathbb{Z}_k^N \to \mathbb{Z}_k^N$.

Such a CA is said to be *linear* if the local update rule is a linear function; that is, if there exist constants $\lambda_{-r}, \ldots, \lambda_r$ such that

$$f(x_{-r}, \ldots, x_r) = \lambda_{-r} x_{-r} + \cdots + \lambda_r x_r \ , \tag{1}$$

where the operations of addition and multiplication are the usual operations of modular arithmetic.

Martin et al [2] study linear CAs by means of polynomials. Denote by R_k^N the set of polynomials with coefficients over $\mathbb{Z}_k$ of degree at most $N-1$. We

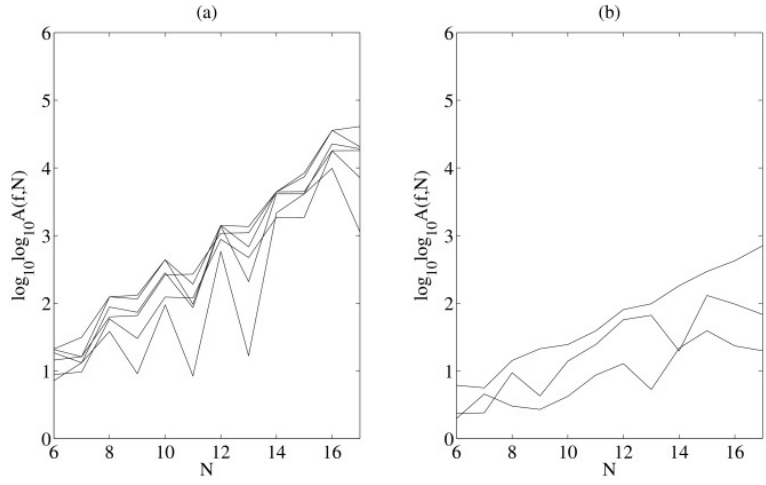

Fig. 4. As Fig. 3, but for $6 \leq N \leq 17$, and showing the two classes of ECAs which do not exhibit a linear relationship between numbers of automorphisms and e^{e^N}. From [1].

can define addition and multiplication in R_k^N similarly to the usual arithmetic of polynomials, but setting $x^N = 1$ (so in effect, powers are computed modulo N). Under these operations, R_k^N is a ring.

Let f be a local update rule of the form of Equation 1. We associate with f the polynomial T_f in R_k^N defined by

$$T_f(x) = \lambda_{-r}x^r + \cdots + \lambda_r x^{-r} \; . \tag{2}$$

Furthermore, we associate with a configuration $s = a_0 a_1 \ldots a_{N-1} \in \mathbb{Z}_k^N$ the polynomial

$$A_s(x) = a_0 + a_1 x + \cdots + a_{N-1}x^{N-1} \; . \tag{3}$$

Then the polynomial associated with the configuration $F(s)$ is simply $T_f(x)A_s(x)$. In other words, repeated application of the global map F corresponds to repeated multiplication by the polynomial $T_f(x)$.

2.1 Rule 90

Let $r = 1$ and $k = 2$, and consider $f : \mathbb{Z}_2^3 \to \mathbb{Z}_2$ defined by

$$f(x_{-1}, x_0, x_1) = x_{-1} + x_1 \; . \tag{4}$$

Since $r = 1$ and $k = 2$, this is an example of an *elementary cellular automaton*. According to Wolfram's numbering scheme [3], f is *rule 90*. The polynomial corresponding to f is

$$T_f(x) = x + x^{-1} \; . \tag{5}$$

Remark 2.1. Let F be the global map for rule 90 on N cells. Choose an initial configuration s_0, and let

$$s_t = \underbrace{F \circ \cdots \circ F}_{t \text{ occurrences}}(s_0) \; . \tag{6}$$

Then at most $O(\log t)$ polynomial multiplications are required to compute s_t.

Proof. It suffices to show that the polynomial $(T_f(x))^t$ can be written as a product of at most $O(\log t)$ polynomials, where each term in this product is either known or computable in constant time.

Since we are working with coefficients over $\mathbb{Z}_2$, we have

$$\left(x + x^{-1}\right)^{2^k} = x^{2^k} + x^{-2^k} \tag{7}$$

for all nonnegative integers k. Thus if t is a power of 2, s_t can be computed by multiplication with $x^t + x^{-t}$.

If t is not a power of 2, it can nevertheless be written as a sum of $\lceil \log_2 t \rceil$ or fewer powers of 2 (i.e. in binary notation). If

$$t = 2^{i_1} + \cdots + 2^{i_l} \; , \tag{8}$$

then

$$\left(x + x^{-1}\right)^t = \left(x + x^{-1}\right)^{2^{i_1}} \ldots \left(x + x^{-1}\right)^{2^{i_l}} \; . \tag{9}$$

The product on the right-hand side involves no more than $\lceil \log_2 t \rceil$ terms, each of which can be determined in constant time via Equation 7. □

2.2 Cycle lengths and the suborder function

For positive integers n and k, the *(multiplicative) suborder function* $\mathrm{sord}_n(k)$ is defined [2] by

$$\mathrm{sord}_n(k) = \begin{cases} \min\left\{ j > 0 \; : \; k^j \equiv \pm 1 \mod n \right\} & \text{if such a } j \text{ exists} \\ 0 & \text{otherwise} \; . \end{cases} \tag{10}$$

Note that $\mathrm{sord}_n(k) \neq 0$ if and only if n and k are relatively prime. In particular, if $k = 2$ then $\mathrm{sord}_n(2)$ is nonzero if n is odd and zero if n is even. The suborder function $\mathrm{sord}_n(2)$ is plotted in Fig. 5.

If n is odd, then we have

$$\log_2 n \leq \mathrm{sord}_n(2) \leq \frac{n-1}{2} \; . \tag{11}$$

The set of values of n for which the upper bound is achieved is a subset of the primes.

Let Π_N denote the length of the cycle reached by rule 90 from an initial configuration which assigns state 1 to a single cell and state 0 to the remainder.

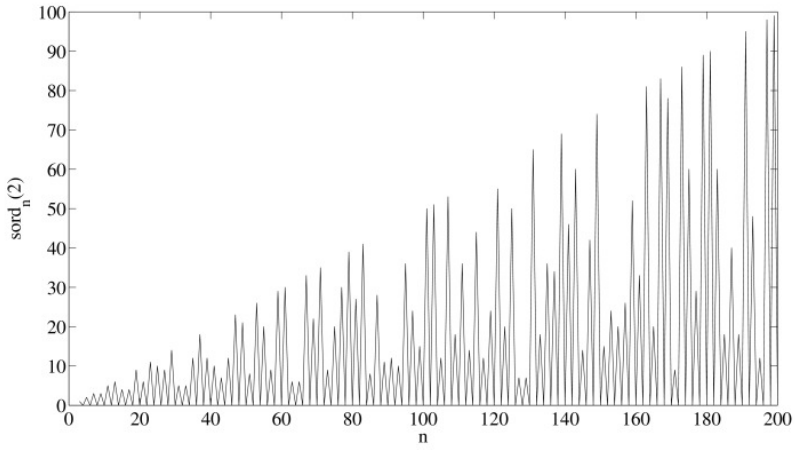

Fig. 5. Plot of the suborder function $\mathrm{sord}_n(2)$ against n, for $3 \leq n \leq 200$.

Due to rule 90's linearity, all cycle lengths must be factors of Π_N. Furthermore, Martin et al [2] show that

$$\Pi_N = \begin{cases} 1 & \text{if } N \text{ is a power of 2} \\ 2\Pi_{N/2} & \text{if } N \text{ is even but not a power of 2} \\ \text{a factor of } 2^{\mathrm{sord}_N(2)} - 1 & \text{if } N \text{ is odd .} \end{cases} \quad (12)$$

3 Counting automorphisms of transition graphs

Definition 3.1. *Consider a CA whose set of configurations is C and whose global map is F. The* transition graph *for this CA is the directed graph with vertex set C and edge set*

$$\{(s, F(s)) \ : \ s \in C\} \ . \quad (13)$$

Every vertex in a transition graph has out-degree 1. This forces the graph to have a "circles of trees" topology: the graph consists of a number of disjoint cycles, with a (possibly single-vertex) tree rooted at each vertex in each cycle.

The *basins* of the transition graph are its disjoint components: each basin consists of exactly one cycle, along with the trees rooted on that cycle.

Examples of transition graphs are shown in Figs. 1 and 2.

Definition 3.2. *Consider a directed graph with vertex set V and edge set E. An* automorphism *on this graph is an isomorphism from the graph to itself; in other words, a bijection $\alpha : V \to V$ such that*

$$(x, y) \in E \iff (\alpha(x), \alpha(y)) \in E \ . \quad (14)$$

Denote by $A(f, N)$ the number of automorphisms in the transition graph for the CA with local rule f on N cells.

In [1] we describe an algorithm for computing the number of automorphisms for a transition graph. This algorithm works by exploiting the recursive structure of the transition graph. For instance, consider a tree, rooted at vertex r, such that the "children" of r are vertices $c_1, \ldots, c_k$. Then the number of automorphisms in the tree is the product of the numbers of automorphisms for each subtree rooted at a c_i, multiplied with the number of permutations of $c_1, \ldots, c_k$ which preserve the isomorphism classes of the subtrees.

Transition graphs for linear CAs have further structure to be exploited:

Lemma 3.1 ([2, Lemma 3.3]). *In a transition graph for a linear CA, the trees rooted at the vertices in the cycles form a single isomorphism class.*

Thus two basins are isomorphic if and only if their cycles have the same length. Cycles of different lengths can occur within a transition graph, so the basins do not necessarily form a single isomorphism class. To find the isomorphism classes, it is necessary (and sufficient) to find the lengths and multiplicities of the cycles. Martin et al [2] give an algorithm for this in rule 90, and it seems reasonable to expect that similar algorithms exist for other linear CAs.

The following results characterise the structure of the trees themselves for rule 90 on N cells:

Theorem 3.1 ([2, Theorem 3.3]). *If N is odd, all trees in the transition graph consist of a single edge.*

Theorem 3.2 ([2, Theorem 3.4]). *If N is even, all trees in the transition graph have the following properties:*

1. *The distance from the root vertex to every leaf vertex is*

$$\frac{1}{2} \max \left\{ 2^j \ : \ 2^j | N \right\}; \tag{15}$$

2. *The root vertex has in-degree 3;*
3. *Every non-root non-leaf vertex has in-degree 4.*

These theorems are illustrated in Figs. 1 and 2.

If N is odd, clearly the only automorphism on each tree is the identity. How many automorphisms does each tree possess if N is even? The following result is an application of [1, Lemma 1].

Lemma 3.2. *Consider a tree of depth $D > 1$, whose root vertex v has in-degree 3 and with all other vertices having in-degree 4. The number of automorphisms for this tree is*

$$A(v) = 24^{2^{2(D-1)}}/4 \, . \tag{16}$$

Proof. See Appendix 7. □

The following theorem is our main result, and follows directly from Lemma 3.2 above and Lemma 2 and Theorem 2 in [1].

Theorem 3.3. *Suppose that, for some value of N, the distinct cycle lengths in rule 90 are $l_1, \ldots, l_k$, and there are m_i cycles of length l_i. Let*

$$A_T = \begin{cases} 1 & \text{if } N \text{ is odd} \\ 24^{2^{N-2}}/4^{2^{N-D_2(N)}} & \text{if } N \text{ is even} , \end{cases} \tag{17}$$

where

$$D_2(N) = \max\left\{2^j \;:\; 2^j | N\right\} . \tag{18}$$

Then

$$A(90, N) = \left(\prod_{i=1}^{k} m_i! \cdot l_i^{m_i} \right) \cdot A_T . \tag{19}$$

Proof. See Appendix 7. □

Thus if the l_is and m_is are known, the number of automorphisms can easily be calculated. The following corollary illustrates a particularly straightforward special case:

Corollary 3.1. *If N is a power of 2, then*

$$A(90, N) = 24^{2^{N-2}}/4 . \tag{20}$$

Proof. See Appendix 7. □

4 Computational results

Martin et al [2] provide cycle lengths and multiplicities for $3 \leq N \leq 40$, as well as an algorithm for computing the lengths and multiplicities for larger N. Using these in conjunction with Theorem 3.3, we are able to compute values of $A(90, N)$ for N much larger than by the general method described in [1]. Some results are shown in Fig. 6.

Compare Fig. 6 with the suborder function $\mathrm{sord}_N(2)$ plotted in Fig. 5. In particular, observe that peaks in one seem to correspond with troughs in the other. Indeed, it can be verified numerically that we have an approximate linear relationship:

$$\log_{10} \log_{10} A(90, N) \approx 0.30N - 0.28\,\mathrm{sord}_N(2) - 0.04 . \tag{21}$$

Figure 7 plots the two sides of Equation 21, and Fig. 8 plots the difference between them against N. Although the correlation is not exact, note that there are no outliers. Also note that the difference between the two sides, and hence the error in this approximation, seems to increase (albeit slowly) with N.

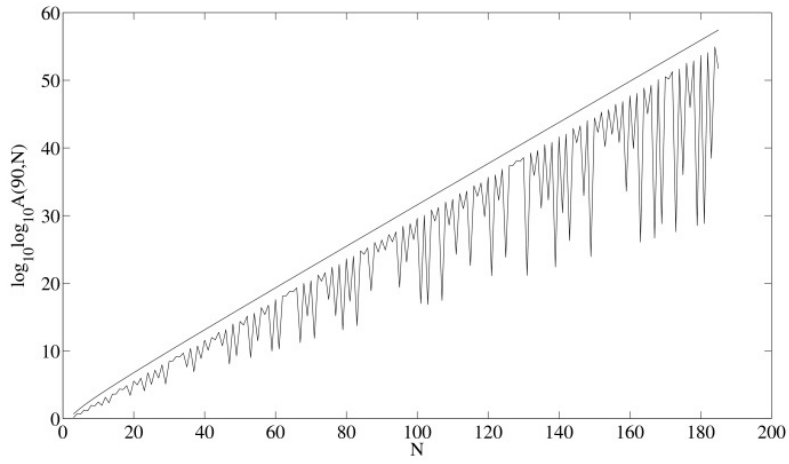

Fig. 6. Plot of $\log_{10} \log_{10} A(90, N)$ (lower line) against N, for $3 \leq N \leq 185$. For comparison, $\log_{10} \log_{10} A(204, N) = 2^N!$ is also plotted (upper line).

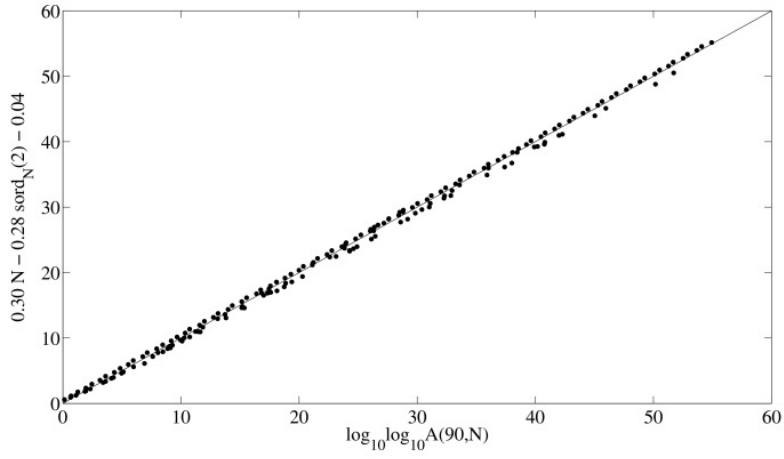

Fig. 7. Plot of the two sides of Equation 21. The diagonal "$y = x$" line is plotted for comparison.

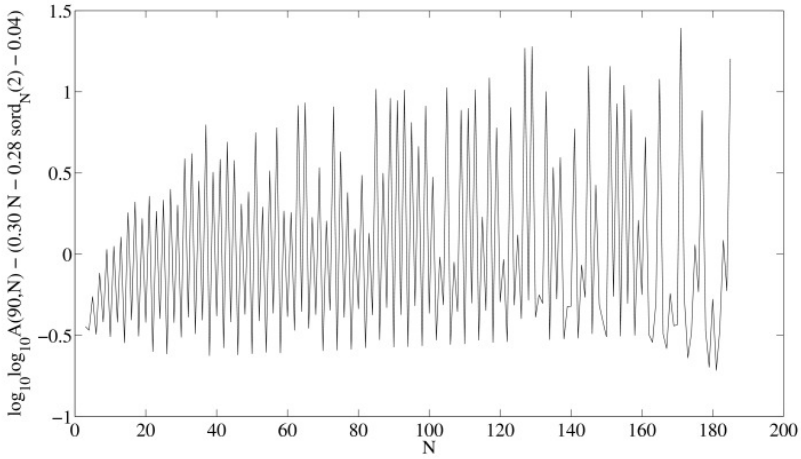

Fig. 8. Plot of the difference between the two sides of Equation 21 against N.

5 Conclusion

Previously [1] we computed numbers of automorphisms for all 88 essentially different ECAs. Implementing the "brute force" method described therein on a current desktop PC, we find that $N = 17$ is the practical limit of what can be computed in a reasonable length of time. Furthermore, the exponential complexity of the computation means that an increase in computational resources would not significantly increase this limit. In contrast, rule 90 has properties which allow for a much more efficient algorithm. On the same desktop PC, we are able to count automorphisms for $N \leq 185$, and for many (though increasingly uncommon) cases beyond, with ease.

However, it seems plausible that there exists an even simpler expression for the number of automorphisms in rule 90, and that the suborder function $\mathrm{sord}_N(2)$ dominates this expression. The suborder function relates to rule 90 since, if N is odd, all cycle lengths must divide $2^{\mathrm{sord}_N(2)} - 1$. It is not clear why the expression

$$\prod_i m_i! \cdot l_i^{m_i} \ , \tag{22}$$

where the l_is are the cycle lengths and the m_is are their respective multiplicities, should be so strongly correlated with this common multiple of the l_is. We plan to investigate this further, and to determine whether this approximate correlation does indeed indicate the existence of a simpler exact expression for $A(90, N)$.

It is reasonable to expect that other linear rules admit a similar approach to that applied here. Indeed, Martin et al [2] generalise some (but not all) of their

results beyond rule 90. We intend to use these more general results to extend our methods to the other linear ECAs, and to other linear CAs in general.

However, these methods almost certainly do not apply to nonlinear CAs: the analogy with finite rings of polynomials is crucial to this work, but this analogy only holds for linear CAs. Thus this work demonstrates (if yet another demonstration were needed!) the ease of analysis and computation for linear CAs as compared to their nonlinear counterparts.

Acknowledgment

This work is funded by an EPSRC DTA award.

6 Appendix

7 Proofs

7.1 Proof of Lemma 3.2

Let u_i be any vertex at depth i in the tree, so $v = u_0$ and u_D is a leaf. Then by [1, Lemma 1], noting that the children of u_i form a single isomorphism class, we have

$$A(v) = A(u_0) = 3!A(u_1)^3 \tag{23}$$
$$A(u_1) = 4!A(u_2)^4 \tag{24}$$
$$\vdots$$
$$A(u_{D-1}) = 4!A(u_D)^4 \tag{25}$$
$$A(u_D) = 1\ . \tag{26}$$

Thus

$$A(v) = 3!\underbrace{(4!(4!(\dots(1)^4\dots)^4)^4)^3}_{D-1 \text{ occurrences of } 4!} \tag{27}$$
$$= 3! \times 4!^3 \times 4!^{3\times 4} \times \cdots \times 4!^{3\times 4^{D-2}} \tag{28}$$
$$= 3! \times 4!^{3\sum_{i=0}^{D-2} 4^i}\ . \tag{29}$$

It can be shown that, for any positive integers n and k,

$$\sum_{i=0}^{n} k^i = \frac{k^{n+1}-1}{k-1}\ . \tag{30}$$

Thus

$$A(v) = 3! \times 4!^{3 \sum_{i=0}^{D-2} 4^i} \tag{31}$$

$$= 3! \times 4!^{3 \times (4^{D-1}-1)/3} \tag{32}$$

$$= \frac{3!}{4!} \times 4!^{4^{D-1}} \tag{33}$$

$$= 24^{2^{2(D-1)}}/4 \tag{34}$$

as required. □

7.2 Proof of Theorem 3.3

[1, Theorem 2] states that

$$A(90, N) = \left(\prod_{I \in \{B_i\}/\cong} |I|! \right) \left(\prod_{i=1}^{k} A(B_i)^{m_i} \right) . \tag{35}$$

By [2, Lemma 3.3], all of the trees rooted at vertices in cycles are isomorphic. Thus two basins are isomorphic if and only if they have the same cycle length, and so we have

$$\prod_{I \in \{B_i\}/\cong} |I|! = \prod_{i=1}^{k} m_i! . \tag{36}$$

Now let $A(B_i)$ be the number of automorphisms for a basin whose cycle length is l_i. By [1, Lemma 2], we have

$$A(B_i) = \frac{l_i}{q} \prod_{j=1}^{l_i} A(v_j) . \tag{37}$$

But all of the trees are isomorphic, so $q = 1$ and thus

$$A(B_i) = l_i A(v)^{l_i} , \tag{38}$$

where $A(v)$ is the number of automorphisms in a tree. Substituting into Equation 35 we have

$$A(90, N) = \left(\prod_{i=1}^{k} m_i! \right) \left(\prod_{i=1}^{k} (l_i A(v)^{l_i})^{m_i} \right) \tag{39}$$

$$= \prod_{i=1}^{k} \left(m_i! \cdot l_i^{m_i} \cdot A(v)^{l_i m_i} \right) \tag{40}$$

$$= \left(\prod_{i=1}^{k} m_i! \cdot l_i^{m_i} \right) \cdot A(v)^{\sum_{i=1}^{k} l_i m_i} . \tag{41}$$

It now suffices to show that

$$A(v)^{\sum_{i=1}^{k} l_i m_i} = A_T \tag{42}$$

with A_T as defined in Equation 17.

If N is odd, [2, Theorem 3.3] states that all trees consist of a single edge. Thus $A(v) = 1$, and so $A(v)^{\sum_{i=1}^{k} l_i m_i} = 1 = A_T$, regardless of the values of $l_i m_i$.

Suppose that N is even. By [2, Theorem 3.4], all trees are of the form described in Lemma 3.2, with $D = D_2(N)/2$. Thus we have

$$A(v) = 24^{2^{D_2(N)-2}}/4 \,. \tag{43}$$

Now, $\sum_{i=1}^{k} l_i m_i$ is simply the number of configurations which occur in cycles, and thus, by a corollary to [2, Theorem 3.4], is given by

$$\sum_{i=1}^{k} l_i m_i = 2^{N-D_2(N)} \,. \tag{44}$$

Hence

$$\begin{aligned} A(v)^{\sum_{i=1}^{k} l_i m_i} &= 24^{2^{D_2(N)-2} \times 2^{N-D_2(N)}}/4^{2^{N-D_2(N)}} && (45)\\ &= 24^{2^{D_2(N)-2+N-D_2(N)}}/4^{2^{N-D_2(N)}} && (46)\\ &= 24^{2^{N-2}}/4^{2^{N-D_2(N)}} && (47)\\ &= A_T && (48) \end{aligned}$$

as required. □

7.3 Proof of Corollary 3.1

If N is a power of 2 then $D_2(N) = N$, so

$$A_T = 24^{2^{N-2}}/4 \tag{49}$$

Furthermore, by [2, Lemmas 3.4 and 3.5], the only possible cycle length is 1; by [2, Lemma 3.7], there is only one such cycle. Thus

$$\begin{aligned} A(90, N) &= \left(1! \cdot 1^1\right) \cdot A_T && (50)\\ &= A_T && (51)\\ &= 24^{2^{N-2}}/4 && (52) \end{aligned}$$

as required. □

References

[1] Powley, E., Stepney, S.: Automorphisms of transition graphs for elementary cellular automata. In: proceedings of Automata 2007: 13th International Workshop on Cellular Automata. (2008) To appear in Journal of Cellular Automata.
[2] Martin, O., Odlyzko, A.M., Wolfram, S.: Algebraic properties of cellular automata. Communications in Mathematical Physics **93** (1984) 219–258.
[3] Wolfram, S.: Statistical mechanics of cellular automata. Reviews of Modern Physics **55**(3) (1983) 601–644.